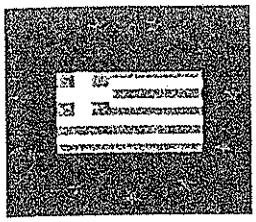
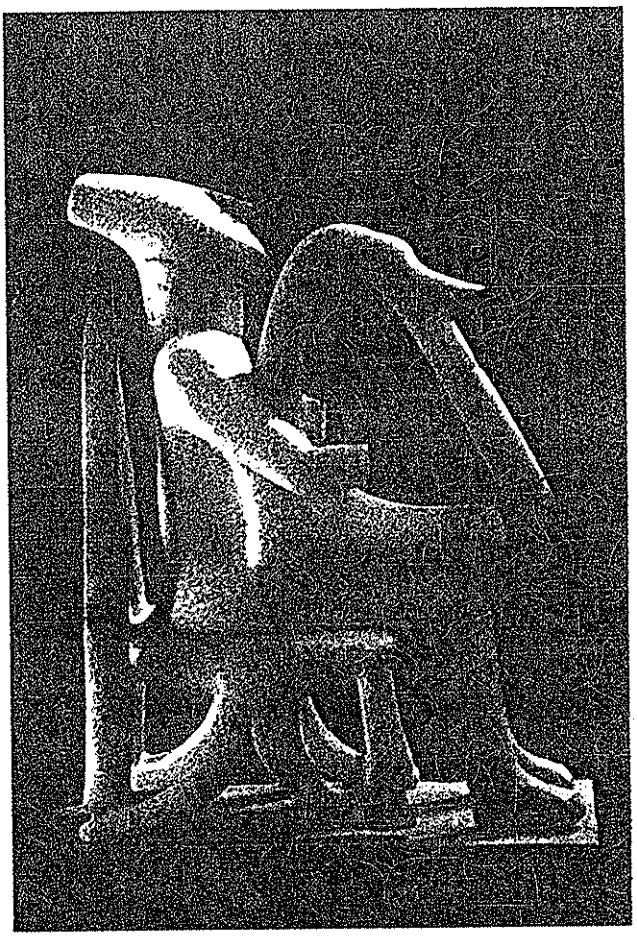


HELLENIC-EUROPEAN RESEARCH ON COMPUTER MATHEMATICS AND ITS APPLICATIONS

HERCMA '98

Proceedings of the Fourth Hellenic-European Conference on Computer Mathematics and its Applications

September 24-26, 1998 - Athens, Hellas



E. A. LIPITAKIS
Editor

Volume 1

L E A

HERCMA '98
E. A. Lipitakis (Editor)
1998, LEA
Printed in Hellas

Multivalued Parallel Recombinative Reinforcement Learning

Aristidis Likas
Department of Computer Science
University of Ioannina
45110, Ioannina, Greece
e-mail: arly@cs.uoi.gr

Abstract

Parallel Recombinative Reinforcement Learning (PRRL) [4, 5] constitutes a population based technique (genetic algorithm) suitable for function optimization in high dimensional binary domains. We present here an extension of this method to the case of population strings whose elements may take any value from a finite discrete set. To achieve this, a new type of stochastic unit (the *multivalued discrete stochastic unit*) is defined that can be used in reinforcement learning problems, and a REINFORCE update scheme for the adaptable parameters of the unit is presented. By considering a population of optimizers of this kind that are appropriately recombined at each step (in a manner analogous to PRRL), we obtain the *Multivalued Parallel Recombinative Reinforcement Learning (MPRRL)* technique and present experimental results from the application of the technique to some test problems.

1 Introduction

Parallel Recombinative Reinforcement Learning (PRRL) [4, 5] constitutes a *population based recombinative* technique for the solution of discrete 0-1 optimization problems. *Population-based* search techniques consider as current state the locations of *many points* in the state space and derive new points by suitably manipulating the current ones. Such techniques have gained much attention mainly since they are suitable for parallel implementation.

A degenerate example of population-based search technique is parallel point-based hill-climbing, where there is a population of point-based hillclimbers operating in parallel and independently, with no information exchange among them. Their exploration capability is limited by the capabilities of individual optimizers. The main interest is focused on *recombinative population based* techniques which generate points for testing by combining information from the current population. The most widely studied optimization procedures of this kind are based on *genetic algorithms* [2] and their hybrids that extend the traditional simple genetic approach by incorporating more sophisticated hillclimbing procedures. The first such

hybrid that has been developed was the SIGH algorithm (Stochastic Iterated Genetic Hill-climbing) [1] and many other attempts followed, such as Parallel Recombinative Simulated Annealing (PRSA) [6] and PRRL [4, 5]. The PRRL method constitutes a population-based technique that tackles discrete optimization problems defined on *binary domains* (i.e. on the unit hypercube). It is based on the parallel operation and combination of individual reinforcement learning optimizers. The approach can be considered as an extension of traditional genetic algorithms in the sense that it considers vectors of probabilities instead of bit vectors as population members. At each step these members are recombined and, in addition, the components of the resulting vectors of probabilities are adjusted according to the reinforcement learning rules. More specifically, the REINFORCE family of algorithms is used that have been proved to follow the stochastic hillclimbing property [8].

It must be noted that no attempt has been previously made to employ reinforcement learning for multivalued discrete optimization problems. For this reason, the PRRL method suffers from the limitation that *it concerns binary population strings* and, consequently, it requires the specification of the discrete optimization problem in a 0-1 formulation. This is natural for several problems, but there are also many cases where the problem variables assume values from finite discrete sets with more than two elements (for example the k-graph coloring problem, where each variable may be assigned any of the k colors). The formulation of multivalued problems as 0-1 problems has two major drawbacks: i) $\sum_{i=1}^{N} k_i$ binary variables are needed for a problem with N variables, where each variable i assumes k_i discrete values and ii) additional constraints must be imposed so that no more that one binary variable corresponding to the same problem variable may be one simultaneously.

In order to avoid the above drawbacks and maintain the multivalued formulation, we propose the *multivalued discrete stochastic (MDS) unit* and a update scheme for the adaptable parameters of the MDS unit based on the REINFORCE theorem. Using a team of MDS units (in a manner analogous to the case of binary stochastic units (called Bernoulli units) [7]), a reinforcement learning optimizer is obtained suitable for treating multivalued problems. Then, following the principles of the PRRL algorithm, we propose the Multivalued PRRL (MPRRL) algorithm that constitutes a population based technique based on the parallel operation and recombination of reinforcement optimizers with MDS units.

2 Multivalued discrete reinforcement learning

In the reinforcement learning approach to function optimization [7, 3] the state of the learning system is determined through a probability distribution. At each step, a point in the function space is generated according to the above distribution, and the corresponding function

value, which is called *reinforcement*, is provided to the system. Then, the parameters of the distribution are updated so as to direct the search towards the generated point in case of a high reinforcement value. In the opposite case, the point is made less probable to be sampled again in the upcoming trials. In order to judge whether a point is 'good' or not, a *standard of comparison* must be specified which in most cases is considered as a trace (weighted average) of past reinforcement values (eq. (4)).

Reinforcement learning schemes have been applied to both continuous and binary [7, 3] function optimization problems. In what concerns the application to problems defined on binary domains, the simplest scheme considers that the point $y = (y_1, \ldots, y_n)$ $(y_j \in \{0, 1\})$ of the problem state space to be evaluated at each step is generated by a team of *Bernoulli units*. Each Bernoulli unit j determines the component y_j of the binary output vector through a Bernoulli selection with probability $p_j = f(w_j)$, where $W = (w_1, \ldots, w_n)$ is the vector of adjustable parameters (weights) and f is a sigmoid function of the form

$$p_j = f(w_j) = 1/(1 + \exp(-w_j)) \tag{1}$$

The MDS unit constitutes an extension of the Bernoulli unit that provides as output y one-out-of M possible values $\{a_1, \ldots, a_M\}$. An MDS unit is characterized by a parameter vector $w = (w_1, \ldots, w_M)$, where each parameter w_i corresponds to the output value a_i. During selection, using the parameter vector w, a probability vector $p = (p_1, \ldots, p_M)$ is computed as follows:

$$p_i = \frac{\exp(w_i/T)}{\sum_{j=1}^{M} \exp(w_j/T)} \tag{2}$$

It is obvious that $\sum_{i=1}^{M} p_i = 1$. By performing selection using the probability vector p one of the M values a_i is selected as the output y of the MDS unit.

In order to use the MDS unit for function optimization the update scheme of the parameters w_i must be specified. In our approach we have selected an update scheme based on the REINFORCE theorem [7, 8].

REINFORCE algorithms [8] constitute an important class of reinforcement learning algorithms. When applied to a team of stochastic units a REINFORCE algorithm prescribes that at each step the weights are updated according to the formula:

$$\Delta w_i = \alpha(r - \bar{r}) \frac{\partial \ln g}{\partial w_i} \tag{3}$$

where α is the learning rate factor, r is the reinforcement signal delivered by the environment and $\bar{r}$ the reinforcement comparison:

$$\bar{r}(t) = \gamma \bar{r}(t-1) + (1 - \gamma) r(t) \tag{4}$$

(γ is a decay rate positive and less than 1). In addition, $g(a,w) = Prob\{y = a|w\}$ is the probability that the output y will be equal to the value a given that the parameter vector is w.

An important result proved in [8] is that, for any REINFORCE algorithm the average update in parameter space W lies in a direction for which the expected value of r is increasing, i.e., the algorithm is characterized by the *stochastic hillclimbing property*. Therefore, REINFORCE algorithms can be used to perform stochastic function maximization [7, 8, 3].

In the case of the MDS unit it holds that

$$\frac{\partial \ln g}{\partial p_i} = \begin{cases} \frac{1}{p_i} & \text{if } y = a_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Moreover,

$$\frac{\partial \ln g}{\partial w_i} = \sum_{k=1}^{M} \frac{\partial \ln g}{\partial p_k} \frac{\partial p_k}{\partial w_i} \tag{6}$$

and from eq. (2)

$$\frac{\partial p_i}{\partial w_k} = \begin{cases} \frac{1}{T} p_i(1-p_i) & \text{if } k = i \\ -\frac{1}{T} p_i p_k & \text{if } k \neq i \end{cases} \tag{7}$$

Using the above equations we finally find that the *in the case where the selected output is* a_k the parameter update equation suggested by the REINFORCE theorem is

$$\Delta w_i = \begin{cases} \alpha(r - \overline{r}) \frac{1}{T} p_i(1-p_i) & \text{if } i = k \\ -\alpha(r - \overline{r}) \frac{1}{T} p_i p_k & \text{if } i \neq k \end{cases} \tag{8}$$

It must be noted that the above pure REINFORCE scheme converges, for this reason a simple modification has been suggested [7] that incorporates a decay term $-\delta w_i$ in the update equation (8) in order to achieve the sustained exploration objective:

$$\Delta w_i = \begin{cases} \alpha(r - \overline{r}) \frac{1}{T} p_i(1-p_i) - \delta w_i & \text{if } i = k \\ -\alpha(r - \overline{r}) \frac{1}{T} p_i p_k - \delta w_i & \text{if } i \neq k \end{cases} \tag{9}$$

where $0 < \delta < 1$. The above equation is the reinforcement update equation used in the MPRRL algorithm.

Using a team of n MDS units, with each unit i ($i = 1, \ldots, n$) having parameter vector w^i and set of output values $D_i = \{d_{i1}, \ldots, d_{ik_i}\}$ (where $k_i = |D_i|$), it is possible to tackle multivalued discrete optimization problems with n variables X_i, each one assuming values from the discrete set D_i ($i = 1, \ldots, n$). The algorithm is simple: at each step each MDS unit i selects the output y_i (using the parameter vector w^i) and thus a point $y = (y_1, \ldots, y_n)$ of the problem state space is obtained. The reinforcement signal r delivered to the learning system is the fitness of the point y and, in the sequent, all MDS units update their parameter vectors w^i using eq. (9).

3 The MPRRL algorithm

Multivalued Parallel Recombinative Reinforcement Learning (MPRRL) can be considered as a population-based recombinative extension of the MDS reinforcement learning approach to multivalued discrete optimization. It is based on the employment of p population members, where each population member i is an MDS reinforcement learning optimizer.

Consider an discrete optimization problems with n variables, where each variable i take values from a finite discrete set $D_i = \{d_{i1}, \ldots, d_{iM}\}$. Without loss of generality, let M denote the size of each set D_i. This means that each member i of the population ($i = 1, \ldots, p$) is a team of n MDS units with weight set w_j^i ($j = 1, \ldots, n$), ie. each member i can be considered as a parameter vector $W_i = (w_1^i, \ldots, w_n^i)$, where $w_j^i = (w_{j1}^i, \ldots, w_{jM}^i)$. In this sense, we can say that each reinforcement optimizer i is assigned one *population slot*, where its parameter vector W_i is kept. At each step, first a reproduction procedure takes place, during which the parameter vectors are recombined and a new generation of vectors is created. Then a sampling procedure is performed and p points $Y_i = (y_{i1}, \ldots, y_{in})$ ($i = 1, \ldots, p$) (with $y_{ij} \in D_j$) of the problem state space are generated using the MDS selection scheme described in the previous section. The fitness r_i (corresponding function value) of each point Y_i is evaluated and a reinforcement update of the parameter vectors W_i takes place using equation (9) so that the search is guided towards promising regions of the space.

At each generation step, the p new population members are created as follows: for each current member i we decide with probability p_c whether crossover will be applied or not. If the decision is negative, the parameter vector of slot i does not change, otherwise, another member k is randomly selected (with probability analogous to its fitness r_k)and crossover is performed between the probability vectors corresponding to the two parents. The recombination operator that we have adopted is a variant of single-point crossover. The new parameter vector W_l is created in the following manner:

- We randomly select the crossover point t ($1 \leq t \leq n-1$).
- If $t \leq \lceil n/2 \rceil$, then we set $w_{lj} = w_{kj}$ for $j = 1, \ldots, t$ and $w_{lj} = w_{ij}$ for $j = t+1, \ldots, n$.
- If $t > \lceil n/2 \rceil$, then we set $w_{lj} = w_{ij}$ for $j = 1, \ldots, t$ and $w_{lj} = w_{kj}$ for $j = t+1, \ldots, n$.

According to the above approach, the new vector W_l remains as close as possible to the vector W_i. That child becomes the new parameter vector W_i for slot i. In this way, the characteristics of individual population members are preserved to some extent, thus retarding the decrease of population diversity. It must be noted that the above reproduction scheme is *synchronous*,

i.e., all p children can be created simultaneously and independently based on the precedent generation, thus achieving a *high degree of parallelism*.

At the beginning, all the components of the parameter vectors are set equal to zero, i.e., no initial knowledge is provided to the optimization system. In case no decay term is used in the reinforcement update rule, the search process converges, in the sense that all probability vectors tend to be similar to each other and, in addition, their individual components tend to be 1 or 0. This convergence behavior is justified by the use of both the crossover mechanism and the reinforcement update rule. The crossover mechanism destroys *population diversity* and eventually all optimizers search the same region of the function space. On the other hand, the local search performed through the use of reinforcement updates eventually converges to a point of high fitness value. If a decay term is added, the search algorithm does not converge, in the sense that it does not continuously produce the same points. Thus, sustained exploration is achieved, but we still have a lack of *sustained diversity* due to the effects of crossover, since the parameter vectors of population members are relatively close.

3.1 Sustained Diversity

In analogy with the PRRL case, in order to reduce the effects of crossover and avoid *genetic drift*, the MPRRL algorithm employs a mechanism based on the notion of *apathy* [1] that has proved very effective in maintaining population diversity.

According to the latter approach, some population members remain apathetic for some generations in the sense that they cannot be selected for recombination. Apathetic members cannot change their state through crossover, but can be chosen for crossover by other members of the population. To effectively apply this principle, a criterion is needed for a member to become apathetic as a well as a criterion for becoming active again. We have chosen to put a member into apathy whenever it generates a point of the state space yielding a higher fitness than the best value achieved so far. Thus, from the moment the search attains a high fitness region, the optimizer is allowed to explore that region following the reinforcement learning rule. If for a specified number of steps no better solution is obtained the member is brought back to the active state. Thus, an apathy step counter is necessary for each population member.

Following the discussion presented above, the MPRRL algorithm has the following final form.

- Initialize all parameter vectors to zero.
- Repeatedly generate a new population from the current one until a maximum number of generations is attained. Each new population is created by performing the following

	MPRRL		MPRL		GA	
No. of nodes	*Succ (%)*	*Avg. Steps*	*Succ (%)*	*Avg. Steps*	*Succ (%)*	*Avg. Steps*
30	100	329	100	421	40	1892
60	100	572	100	654	16.7	3512
90	100	1102	100	1356	6.7	
120	100	1543	90	1778	0	

Table 1: Comparative results for the 3-graph coloring problem

steps for each location $i = 1, \ldots, p$:

1. If member i is in apathy proceed to Step 4.
2. With probability p_c decide whether crossover will be performed or not. If the decision is negative proceed to Step 4.
3. Randomly choose a member from the rest of the population with probability analogous to its fitness. Combine the two parents to produce a new parameter vector as described previously. The new vector replaces the current one in location i.
4. Based on the created parameter vector, generate a point of the state space using the MDS approach and evaluate its fitness r_i.
5. Update the parameters W_i of location i according to the reinforcement learning rule.
6. If $r_i > r_i^{\max}$ set $r_i^{\max} = r_i$. If moreover the population member i is not in apathy then put it into apathy and proceed to Step 8.
7. If the member is in apathy increase the value of its counter by 1. If the value of the counter is the maximum allowed put the member back into the active state and set the counter to zero.
8. Update the value of reinforcement comparison $\overline{r}_i$.

4 Experimental results and Comparisons

Experiments have been conducted on several multivalued discrete optimization problems to compared the effectiveness of MPRRL against the traditional genetic algorithm (GA) and the MPRL (multivalued parallel reinforcement learning) approach, which considers a population of independent multivalued reinforcement learning optimizers. The latter was tested in order to study the effectiveness of recombination. We report here results concerning the NP-complete 3-graph coloring problem, on difficult graph instances, concerning sparse graphs

with a very small number of solutions. For each graph size, 30 experiments were performed using each technique. The maximum allowed number of generations was 5000, while the population size was 100. The values of the parameters of the MPRRL and MPRL methods were $\alpha = 0.1$, $\delta = 0.002$ and $T = 1.0$ and the crossover probability for MPRRL was $p_c = 1.0$. Also the number of apathetic steps was 150. The parameters of the GA were: crossover probability $p_c = 0.6$ and mutation probability $p_m = 0.01$. Average values concerning the percentage of runs that provided a problem solution and the average number of generation steps required to find the solution are displayed Table 1. It is clear that the MPRRL method is much more effective compared with the other approaches. It must also be noted that the MPRL approach also yielded encouraging results, although of lower quality than the MPRRL method.

References

[1] Ackley, D. H., *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers, 1987.

[2] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[3] Kontoravdis, D., Likas, A. and Stafylopatis, A. *A Reinforcement Learning Algorithm for Networks of Units with Two Stochastic Levels*, Journal of Intelligent Systems, vol. 5, no. 1, pp. 50-76, 1995.

[4] Likas, A., Blekas, K. and Stafylopatis, A., *Parallel Recombinative Reinforcement Learning*, Proc. Int. Conf. on Machine Learning (ICML'95), Heraklion, Greece, April 1995, (Lecture Notes in Artificial Intelligence, Vol. 912, pp. 311-314, Springer Verlag, 1995).

[5] Likas, A., Blekas, K. and Stafylopatis, A., *Parallel Recombinative Reinforcement Learning: A Genetic Algorithm*, Journal of Intelligent Systems, vol. 6, no. 2, pp. 145-169, 1996.

[6] Mahfoud, S. W. and Goldberg, D. E., *Parallel Recombinative Simulated Annealing: A Genetic Algorithm*, Parallel Computing, vol. 21, pp. 1-28, 1995.

[7] Williams, R.J. and Peng, J., *Function Optimization Using Connectionist Reinforcement Learning Networks*, Connection Science, vol. 3, pp. 241-268, 1991.

[8] Williams, R.J., *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*, Machine Learning, vol. 8, pp. 229-256, 1992.